Loading Data

In [92]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remoun
t, call drive.mount("/content/drive", force_remount=True).
```

In [93]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pylab import rcParams
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [94]:
```
path = "/content/drive/MyDrive/project/AirQualityUCI.csv"
df = pd.read_csv(path)
df #Preview Air Quality Dataset
```

Out[94]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)
0	3/10/2004	18:00:00	2.6	1360	150	11.9	1046
1	3/10/2004	19:00:00	2.0	1292	112	9.4	955
2	3/10/2004	20:00:00	2.2	1402	88	9.0	939
3	3/10/2004	21:00:00	2.2	1376	80	9.2	948
4	3/10/2004	22:00:00	1.6	1272	51	6.5	836
...	...	...	...	...	...	...	...
9352	4/4/2005	10:00:00	3.1	1314	-200	13.5	1101
9353	4/4/2005	11:00:00	2.4	1163	-200	11.4	1027
9354	4/4/2005	12:00:00	2.4	1142	-200	12.4	1063
9355	4/4/2005	13:00:00	2.1	1003	-200	9.5	961
9356	4/4/2005	14:00:00	2.2	1071	-200	11.9	1047

9357 rows × 15 columns

Classification

In [95]:
```
df.shape
```

Out[95]:
```
(9357, 15)
```

In [96]:
```
df.head(100)
```

Out[96]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	N(
0	3/10/2004	18:00:00	2.6	1360	150	11.9	1046	
1	3/10/2004	19:00:00	2.0	1292	112	9.4	955	
2	3/10/2004	20:00:00	2.2	1402	88	9.0	939	
3	3/10/2004	21:00:00	2.2	1376	80	9.2	948	
4	3/10/2004	22:00:00	1.6	1272	51	6.5	836	
...	...	...	...	...	...	...	...	
95	3/14/2004	17:00:00	2.9	1438	156	12.0	1051	
96	3/14/2004	18:00:00	2.5	1478	122	12.2	1055	
97	3/14/2004	19:00:00	4.6	1808	262	20.6	1312	
98	3/14/2004	20:00:00	5.9	1898	341	23.1	1381	
99	3/14/2004	21:00:00	3.4	1560	214	14.7	1140	

100 rows × 15 columns

In [97]:
```
df.isnull()
```

Out[97]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(G
0	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	
9352	False	False	False	False	False	False	False	Fals
9353	False	False	False	False	False	False	False	Fals
9354	False	False	False	False	False	False	False	Fals
9355	False	False	False	False	False	False	False	Fals
9356	False	False	False	False	False	False	False	Fals

9357 rows × 15 columns

In [98]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           9357 non-null   object 
 1   Time           9357 non-null   object 
 2   CO(GT)         9357 non-null   float64
 3   PT08.S1(CO)    9357 non-null   int64  
 4   NMHC(GT)       9357 non-null   int64  
 5   C6H6(GT)       9357 non-null   float64
 6   PT08.S2(NMHC)  9357 non-null   int64  
 7   NOx(GT)        9357 non-null   int64  
 8   PT08.S3(NOx)   9357 non-null   int64  
 9   NO2(GT)        9357 non-null   int64  
 10  PT08.S4(NO2)   9357 non-null   int64  
 11  PT08.S5(O3)    9357 non-null   int64  
 12  T              9357 non-null   float64
 13  RH             9357 non-null   float64
 14  AH             9357 non-null   float64
dtypes: float64(5), int64(8), object(2)
memory usage: 1.1+ MB
```

In [99]:

```
df.tail()
```

Out[99]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	N
9352	4/4/2005	10:00:00	3.1	1314	-200	13.5	1101	
9353	4/4/2005	11:00:00	2.4	1163	-200	11.4	1027	
9354	4/4/2005	12:00:00	2.4	1142	-200	12.4	1063	
9355	4/4/2005	13:00:00	2.1	1003	-200	9.5	961	
9356	4/4/2005	14:00:00	2.2	1071	-200	11.9	1047	

In [100]:

```
df.describe()
```

Out[100]:

	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT
count	9357.000000	9357.000000	9357.000000	9357.000000	9357.000000	9357.00000
mean	-34.207524	1048.990061	-159.090093	1.865683	894.595276	168.61697
std	77.657170	329.832710	139.789093	41.380206	342.333252	257.43386
min	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.00000
25%	0.600000	921.000000	-200.000000	4.000000	711.000000	50.00000
50%	1.500000	1053.000000	-200.000000	7.900000	895.000000	141.00000
75%	2.600000	1221.000000	-200.000000	13.600000	1105.000000	284.00000
max	11.900000	2040.000000	1189.000000	63.700000	2214.000000	1479.00000

In [101]:
```
df.columns
```

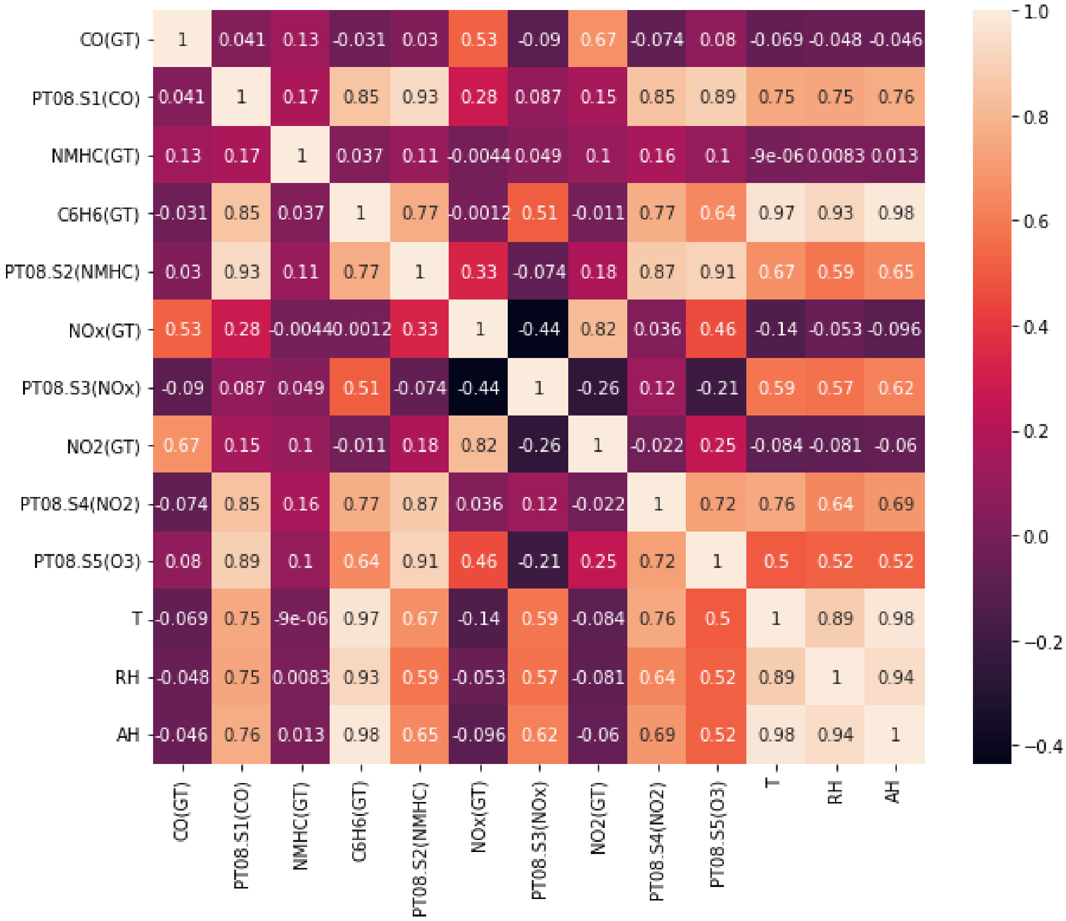
Out[101]:
```
Index(['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(G
T)',
       'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4
(NO2)',
       'PT08.S5(O3)', 'T', 'RH', 'AH'],
      dtype='object')
```

In [102]:
```
df.isnull().sum()
```

Out[102]:
```
Date             0
Time             0
CO(GT)           0
PT08.S1(CO)      0
NMHC(GT)         0
C6H6(GT)         0
PT08.S2(NMHC)    0
NOx(GT)          0
PT08.S3(NOx)     0
NO2(GT)          0
PT08.S4(NO2)     0
PT08.S5(O3)      0
T                0
RH               0
AH               0
dtype: int64
```

In [103]:
```python
plt.figure(figsize=(10,8))
corr=df.corr()
sns.heatmap(corr, annot = True)
```

Out[103]: <matplotlib.axes._subplots.AxesSubplot at 0x7fca5ec5d430>



In [104]:
```python
features = df.columns.tolist()[2:]
features
```

Out[104]:
```
['CO(GT)',
 'PT08.S1(CO)',
 'NMHC(GT)',
 'C6H6(GT)',
 'PT08.S2(NMHC)',
 'NOx(GT)',
 'PT08.S3(NOx)',
 'NO2(GT)',
 'PT08.S4(NO2)',
 'PT08.S5(O3)',
 'T',
 'RH',
 'AH']
```

In [105]:
```
X = df[features].drop('AH',1)
X
```

Out[105]:

	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(N
0	2.6	1360	150	11.9	1046	166	1(
1	2.0	1292	112	9.4	955	103	1
2	2.2	1402	88	9.0	939	131	1
3	2.2	1376	80	9.2	948	172	1(
4	1.6	1272	51	6.5	836	131	1:
...	...	...	...	...	...	...	
9352	3.1	1314	-200	13.5	1101	472	
9353	2.4	1163	-200	11.4	1027	353	
9354	2.4	1142	-200	12.4	1063	293	
9355	2.1	1003	-200	9.5	961	235	
9356	2.2	1071	-200	11.9	1047	265	

9357 rows × 12 columns

In [106]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for i in df.columns:
  if df[i].dtypes=="object":
    df[i] = le.fit_transform(df[i])

df.head()
```

Out[106]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)
0	152	9	2.6	1360	150	11.9	1046	166
1	152	10	2.0	1292	112	9.4	955	103
2	152	12	2.2	1402	88	9.0	939	131
3	152	13	2.2	1376	80	9.2	948	172
4	152	14	1.6	1272	51	6.5	836	131

In [107]:
```
C = df.astype('int')
Y = C['AH']
Y.shape
```

Out[107]: (9357,)

In [108]:
```
print(X.shape,Y.shape)
```

```
(9357, 12) (9357,)
```

In [109]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
(6549, 12) (6549,)
(2808, 12) (2808,)
```

In [110]:
```
rfc = RandomForestClassifier(n_estimators=1000)
rfc.fit(X_train,Y_train)
```

Out[110]:
```
RandomForestClassifier(n_estimators=1000)
```

In [111]:
```
rfc.feature_importances_
```

Out[111]:
```
array([0.01930937, 0.044958  , 0.04020474, 0.04580561, 0.04818777,
       0.02950794, 0.06783306, 0.04363222, 0.17395818, 0.042874  ,
       0.29849471, 0.1452344 ])
```

In [112]:
```
df.columns
```

Out[112]:
```
Index(['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(G
T)',
       'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4
(NO2)',
       'PT08.S5(O3)', 'T', 'RH', 'AH'],
      dtype='object')
```

In [113]:
```
pred = rfc.predict(X_test)
pred
```

Out[113]:
```
array([1, 0, 1, ..., 0, 0, 1])
```

In [114]:
```
pred.shape
```

Out[114]:
```
(2808,)
```

In [115]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test,pred)
cm
```

Out[115]:
```
array([[ 121,    0,    0,    0],
       [   0, 1270,   59,    0],
       [   0,   49, 1289,    0],
       [   0,    0,   17,    3]])
```

In [116]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,pred)
```

Out[116]:
```
0.9554843304843305
```

In [117]:
```
from sklearn.model_selection import cross_val_score
cross_val_score(rfc,X_train, Y_train, cv=10)
```

Out[117]:
```
array([0.95572519, 0.9389313 , 0.96183206, 0.94198473, 0.95877863,
       0.95419847, 0.95114504, 0.95114504, 0.96030534, 0.95565749])
```

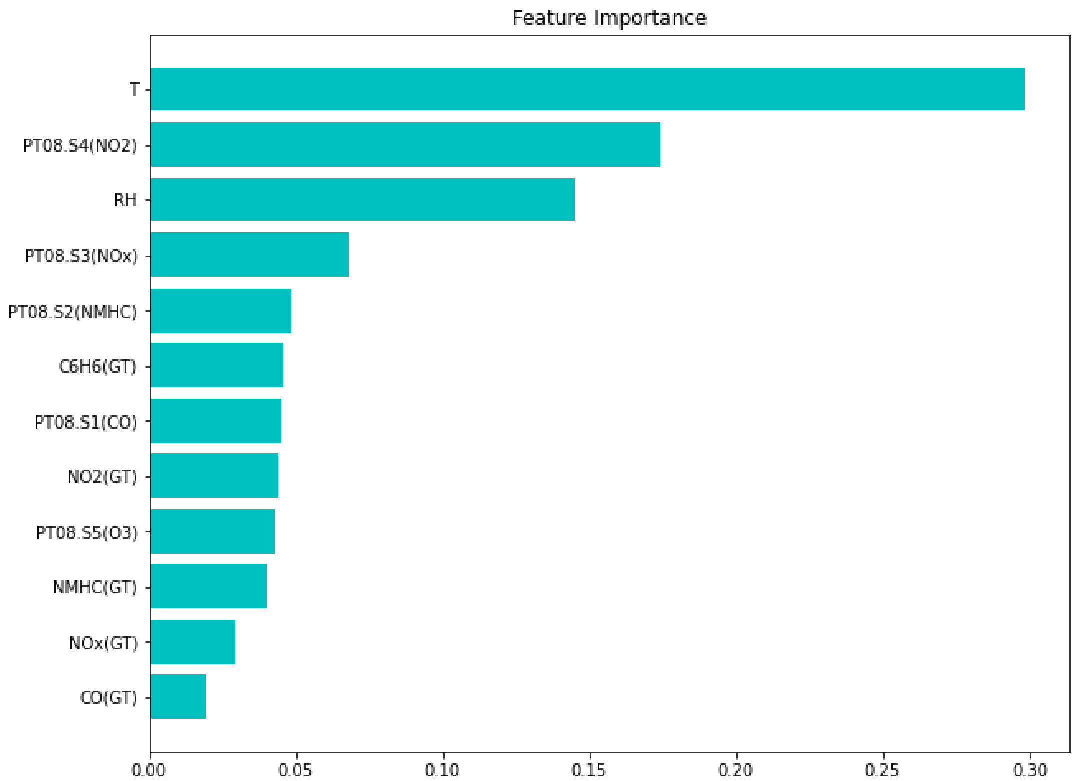
In [118]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test,pred))
```

```
              precision    recall  f1-score   support

        -200       1.00      1.00      1.00       121
           0       0.96      0.96      0.96      1329
           1       0.94      0.96      0.95      1338
           2       1.00      0.15      0.26        20

    accuracy                           0.96      2808
   macro avg       0.98      0.77      0.79      2808
weighted avg       0.96      0.96      0.95      2808
```

In [119]:
```
from numpy.ma.core import indices
importances = rfc.feature_importances_
indices = np.argsort(importances)
plt.figure(figsize=(10,8))
plt.barh(range(len(indices)), importances[indices], color='c',align='ce
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.title('Feature Importance')
plt.show()
```



Regression

In [120]:
```
df.shape #display the no of rows and columns in data set
```

Out[120]:
```
(9357, 15)
```

In [121]:
```
df.head(100) #Display the first 100
```

Out[121]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)
0	152	9	2.6	1360	150	11.9	1046	166
1	152	10	2.0	1292	112	9.4	955	103
2	152	12	2.2	1402	88	9.0	939	131
3	152	13	2.2	1376	80	9.2	948	172
4	152	14	1.6	1272	51	6.5	836	131
...	...	...	...	...	...	...	...	...
95	160	8	2.9	1438	156	12.0	1051	180
96	160	9	2.5	1478	122	12.2	1055	160
97	160	10	4.6	1808	262	20.6	1312	261
98	160	12	5.9	1898	341	23.1	1381	325
99	160	13	3.4	1560	214	14.7	1140	217

100 rows × 15 columns

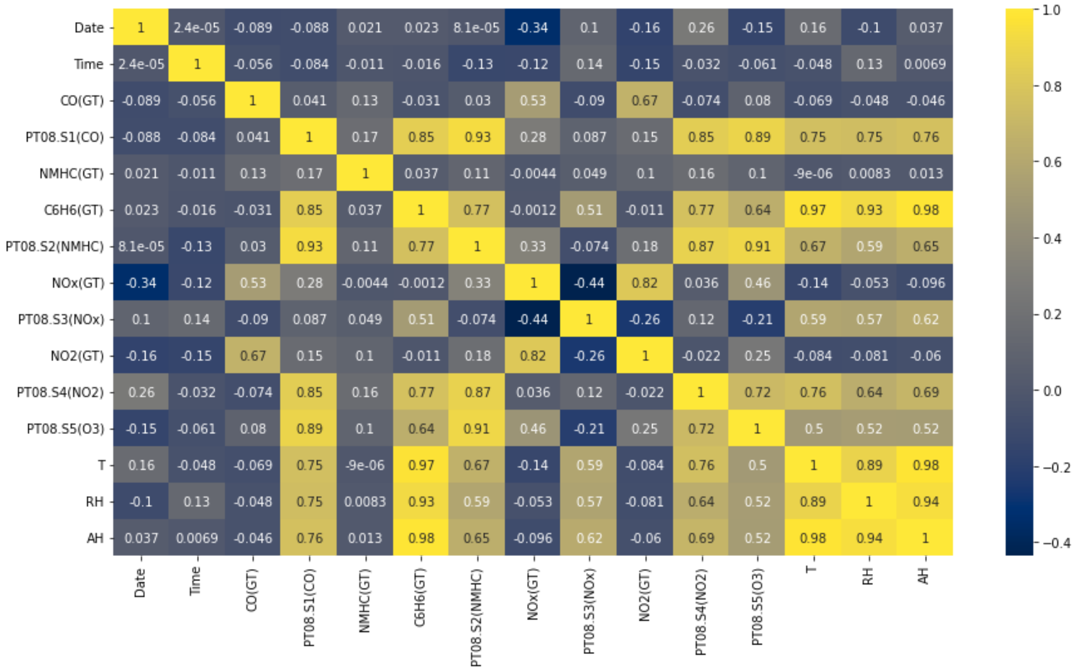
In [122]:
```
df.info() # display the information related to dataset like (null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           9357 non-null   int64  
 1   Time           9357 non-null   int64  
 2   CO(GT)         9357 non-null   float64
 3   PT08.S1(CO)    9357 non-null   int64  
 4   NMHC(GT)       9357 non-null   int64  
 5   C6H6(GT)       9357 non-null   float64
 6   PT08.S2(NMHC)  9357 non-null   int64  
 7   NOx(GT)        9357 non-null   int64  
 8   PT08.S3(NOx)   9357 non-null   int64  
 9   NO2(GT)        9357 non-null   int64  
 10  PT08.S4(NO2)   9357 non-null   int64  
 11  PT08.S5(O3)    9357 non-null   int64  
 12  T              9357 non-null   float64
 13  RH             9357 non-null   float64
 14  AH             9357 non-null   float64
dtypes: float64(5), int64(10)
memory usage: 1.1 MB
```

In [123]:

```python
rcParams['figure.figsize']=(15,8)  #annot -> an array of the same shape
sns.heatmap(df.corr(),annot=True, cmap = 'cividis')  #to check the core
```

Out[123]: <matplotlib.axes._subplots.AxesSubplot at 0x7fca5cb8aeb0>



In [124]:

```python
df1 = df.loc[(df['T'] > 23) & (df['T'] < 24)]
df1
```

Out[124]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT
115	162	4	2.9	1417	207	14.9	1146	17
116	162	5	2.9	1400	191	15.4	1162	15
119	162	8	2.8	1445	214	14.8	1141	15
144	164	9	3.4	1447	237	17.8	1235	18
169	166	10	7.6	1973	577	38.4	1737	41
...	...	...	...	...	...	...	...	..
9140	186	5	1.3	1085	-200	5.1	769	15
9141	186	6	1.3	1177	-200	7.9	896	16
9187	190	4	1.4	1029	-200	3.3	670	15
9188	190	5	1.2	1071	-200	4.8	752	13
9234	195	3	1.9	1136	-200	9.6	962	26

290 rows × 15 columns

Linear Regression

In [125]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error  #to
```

In [126]:

```
col=df1.columns.tolist()[2:]
X=df1[col].drop('RH',1)     #X-input features
print(X)                  #y-input features
```

```
      CO(GT)  PT08.S1(CO)  NMHC(GT)  C6H6(GT)  PT08.S2(NMHC)  NOx(GT)  \
115      2.9         1417       207      14.9           1146      171
116      2.9         1400       191      15.4           1162      159
119      2.8         1445       214      14.8           1141      156
144      3.4         1447       237      17.8           1235      184
169      7.6         1973       577      38.4           1737      411
...      ...          ...       ...       ...            ...      ...
9140     1.3         1085      -200       5.1            769      156
9141     1.3         1177      -200       7.9            896      169
9187     1.4         1029      -200       3.3            670      154
9188     1.2         1071      -200       4.8            752      132
9234     1.9         1136      -200       9.6            962      265

      PT08.S3(NOx)  NO2(GT)  PT08.S4(NO2)  PT08.S5(O3)     T      AH
115            830      119          1831         1404  23.3  0.9096
116            838      111          1829         1263  23.9  0.8757
119            857      110          1824         1252  23.8  0.9137
144            859      139          1778         1296  23.9  0.7519
169            617      194          2414         2306  23.1  0.7403
...            ...      ...           ...          ...   ...     ...
9140           711       85          1264          820  23.7  1.1366
9141           616       94          1374         1011  23.1  1.1523
9187           800       87          1223          678  23.1  1.1506
9188           736       81          1289          668  23.6  1.1320
9234           594      130          1350         1052  23.6  0.9986

[290 rows x 12 columns]
```

In [127]:

```
Y=df1[['RH']]
print(Y.shape)
print(Y)
```

```
(290, 1)
        RH
115   32.2
116   30.0
119   31.3
144   25.7
169   26.5
...    ...
9140  39.4
9141  41.2
9187  41.3
9188  39.4
9234  34.8

[290 rows x 1 columns]
```

In [128]:
```
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,test_size=0.3, ra
lr = LinearRegression()
method=lr.fit(X_train,Y_train)
method
```

Out[128]:
```
LinearRegression()
```

In [129]:
```
print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)
```

```
(203, 12) (203, 1)
(87, 12) (87, 1)
```

In [130]:
```
print('Intercept:',method.intercept_)
print('Slope:')
list(zip(X.columns.tolist(),method.coef_))
```

```
Intercept: [65.45227708]
Slope:
```

Out[130]:
```
[('CO(GT)',
  array([ 5.57910524e-05, -7.48491353e-05,  9.37717278e-05, -2.9326146
9e-02,
          7.89538845e-04,  2.49416918e-04,  1.86431986e-04, -5.2637083
9e-04,
          3.18281110e-04, -8.45348032e-05, -2.81733277e+00,  3.4856746
2e+01]))]
```

In [131]:

```
Y_pred = lr.predict(X_test)
Y_pred
```

Out[131]:
```
array([[46.18074685],
       [57.71870774],
       [29.00344967],
       [55.50949218],
       [63.18949238],
       [63.8087211 ],
       [29.34929775],
       [47.34681429],
       [30.41197254],
       [47.06337823],
       [61.69189887],
       [60.74610137],
       [30.65168166],
       [35.80143346],
       [53.46448893],
       [38.8478863 ],
       [30.49719875],
       [27.95355679],
       [54.50050509],
       [62.16057164],
       [69.49701785],
       [39.25647342],
       [27.69870207],
       [43.18505741],
       [47.40874584],
       [51.45849307],
       [49.89905671],
       [34.80661818],
       [45.57179837],
       [28.62805561],
       [14.02934122],
       [44.60102385],
       [23.28508191],
       [38.51867483],
       [57.0955    ],
       [54.2064708 ],
       [33.03190455],
       [36.7853549 ],
       [41.70839468],
       [32.76106624],
       [61.33594575],
       [35.11953665],
       [65.31368212],
       [49.19978252],
       [51.522501  ],
       [31.46450593],
       [29.92306114],
       [36.2933684 ],
       [26.64506587],
       [46.55868486],
       [64.78599824],
       [39.66094559],
       [37.01420839],
       [54.51843761],
       [42.79171121],
       [53.83531775],
       [48.94665302],
       [46.63003728],
       [27.35706972],
       [43.26022149],
       [27.54939613],
```

```
       [63.6466169 ],
       [48.22469778],
       [34.0081318 ],
       [38.39795009],
       [61.45919855],
       [43.49317861],
       [39.79862297],
       [37.09490786],
       [40.77607815],
       [29.10390683],
       [28.49830661],
       [59.19302453],
       [39.67617633],
       [42.9197693 ],
       [29.68611482],
       [30.89574454],
       [41.28828584],
       [69.55096287],
       [60.97172957],
       [42.09915799],
       [39.11707659],
       [64.43555513],
       [63.72047282],
       [38.68476546],
       [40.67992711],
       [67.49389233]])
```

In [132]:

```
Y_pred = pd.DataFrame(Y_pred, columns = ['RH'])
Y_pred
```

Out[132]:

	RH
0	46.180747
1	57.718708
2	29.003450
3	55.509492
4	63.189492
...	...
82	64.435555
83	63.720473
84	38.684765
85	40.679927
86	67.493892

87 rows × 1 columns

In [133]:

```
mean_sq=np.sqrt(mean_squared_error(Y_test,Y_pred))
print('Root Mean Squared Error:',mean_sq)
```

```
Root Mean Squared Error: 0.22532948202734948
```

In [134]:
```
print('Mean Squared Error:',mean_squared_error(Y_test, Y_pred))
```

Mean Squared Error: 0.050773375470713616

In [135]:
```
print('Mean Absolute Error:',mean_absolute_error(Y_test, Y_pred))
```

Mean Absolute Error: 0.18181574423236183

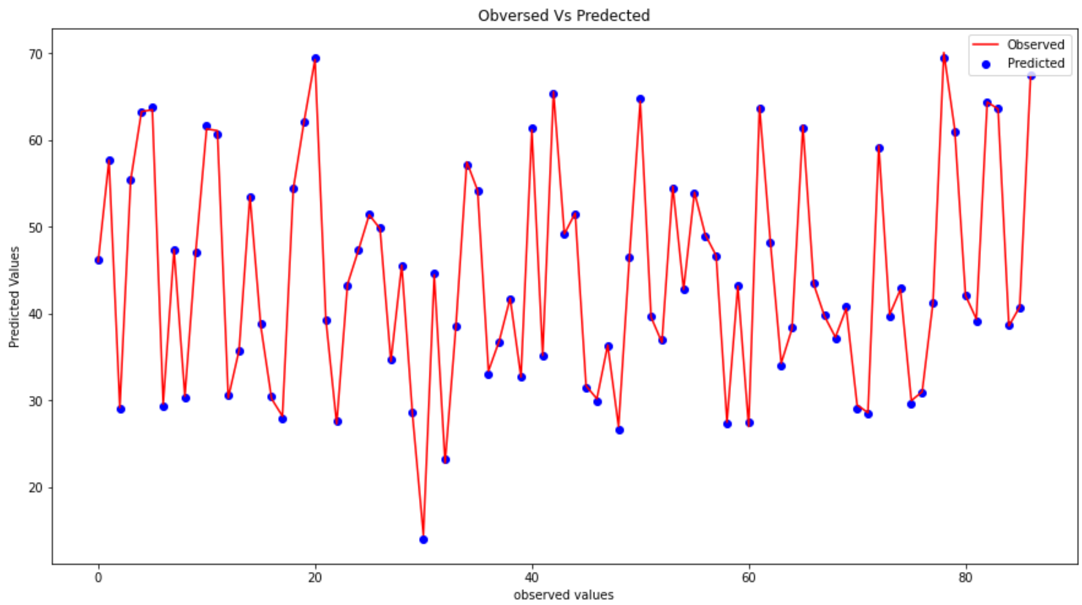
In [136]:
```
print('Testing Accuracy:',lr.score(X_test, Y_test)*100)  #Score between
```

Testing Accuracy: 99.96883240890604

In [137]:
```
print('Training Accuracy:',lr.score(X_train, Y_train)*100) #Score betwe
```

Training Accuracy: 99.97385551860269

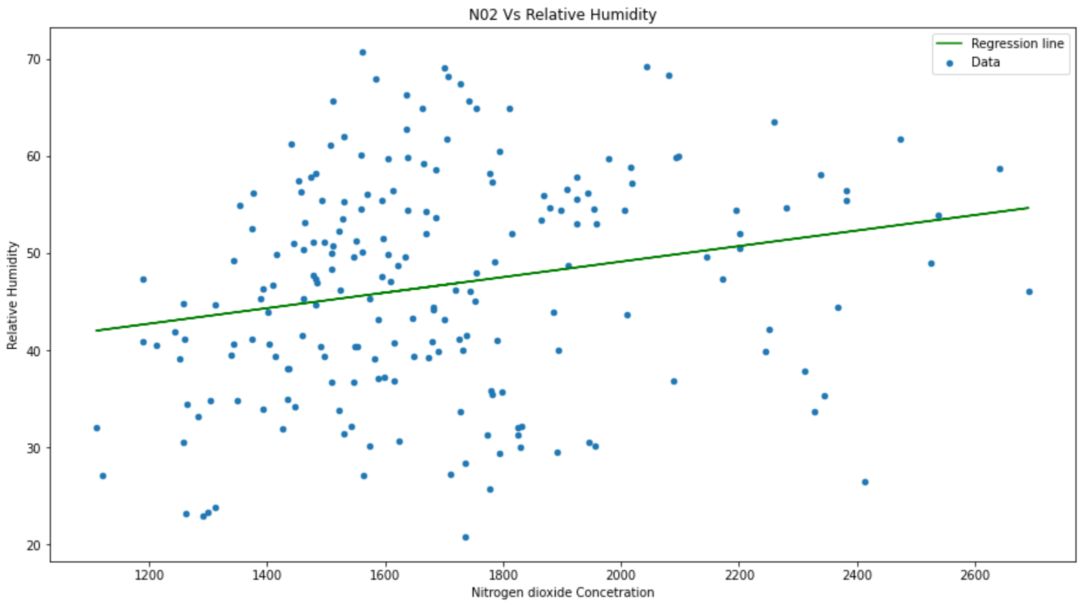
In [138]:
```
plt.rcParams['figure.figsize'] = (15,8)
x_axis = range(len(X_test))
plt.plot(x_axis, Y_test, label = 'Observed', color = 'r')
plt.scatter(x_axis, Y_pred, label = 'Predicted', color = 'b')
plt.title('Obversed Vs Predected')
plt.xlabel('observed values')
plt.ylabel('Predicted Values')
plt.legend()
plt.show()
plt.savefig('/content/drive/MyDrive/project/scatter.png')
```



<Figure size 1080x576 with 0 Axes>

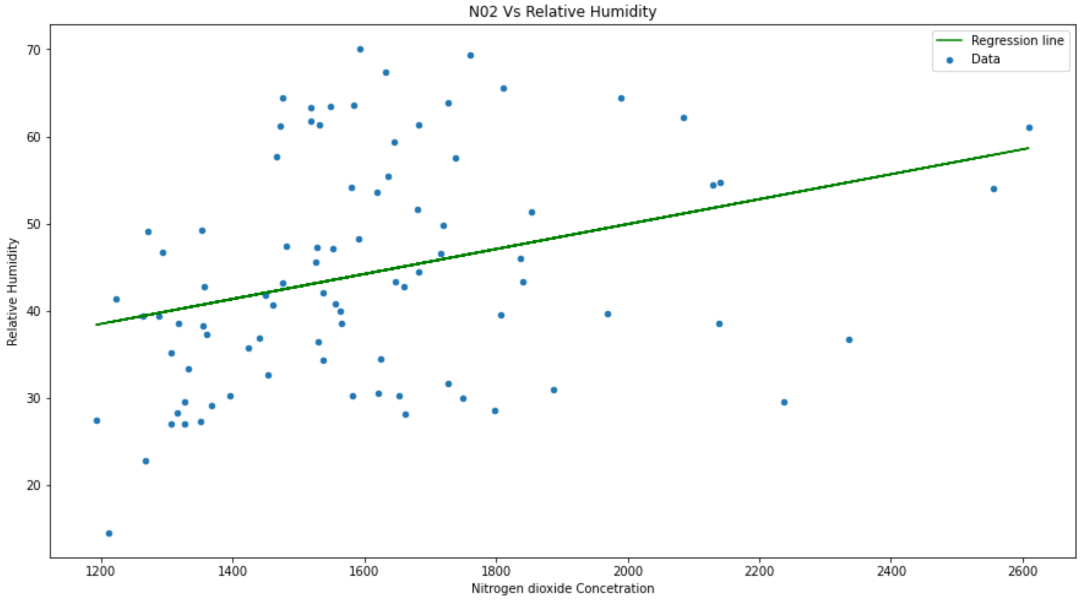
In [139]:

```
#plot for train data
x1 = X_train[['PT08.S4(NO2)']]
lr.fit(x1,Y_train)
Y = lr.predict(x1)
plt.rcParams['figure.figsize'] = (15,8)
plt.scatter(x1, Y_train, s = 20, label = 'Data')
plt.plot(x1, Y, color = 'green', label = 'Regression line')
plt.title('N02 Vs Relative Humidity')
plt.xlabel('Nitrogen dioxide Concetration')
plt.ylabel('Relative Humidity')
plt.legend()
plt.show()
plt.savefig('/content/drive/MyDrive/project/scatter.png')
```



<Figure size 1080x576 with 0 Axes>

In [140]:
```
#plot for test data
x2 = X_test[['PT08.S4(NO2)']]
lr.fit(x2,Y_test)
Y = lr.predict(x2)
plt.rcParams['figure.figsize'] = (15,8)
plt.scatter(x2, Y_test, s = 20, label = 'Data')
plt.plot(x2, Y, color = 'green', label = 'Regression line')
plt.title('N02 Vs Relative Humidity')
plt.xlabel('Nitrogen dioxide Concetration')
plt.ylabel('Relative Humidity')
plt.legend()
plt.show()
```



Random Forest Regression

In [141]:
```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor()
rf
```

Out[141]: RandomForestRegressor()

In [142]:
```
rf_method=rf.fit(X_train,Y_train)          #fit model
Y_pred_rf=rf_method.predict(X_test)            #predict
```

In [143]:
```
print('Root Mean Squared Error:',np.sqrt(mean_squared_error(Y_test,Y_pr
```

Root Mean Squared Error: 1.2499233953538365

In [144]:
```
print('Mean Squared Error:',mean_squared_error(Y_test, Y_pred_rf))
```

Mean Squared Error: 1.5623084942528631

In [145]:
```
print('Mean Absolute Error:',mean_absolute_error(Y_test, Y_pred_rf))
```

Mean Absolute Error: 0.714678160919536

In [146]:
```
result = pd.DataFrame(columns=['Model', ' Mean Squared Error', 'Root Me
result.loc[0] = ['Linear Regression', mean_squared_error(Y_test, Y_pred
result.loc[1] = ['Random Forest Regression', mean_squared_error(Y_test,
result
```

Out[146]:

	Model	Mean Squared Error	Root Mean Squared Error	Mean Absolute Error
0	Linear Regression	0.050773	0.225329	0.181816
1	Random Forest Regression	1.562308	1.249923	0.714678

Clustering

In [147]:
```
df
```

Out[147]:

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT
0	152	9	2.6	1360	150	11.9	1046	16(
1	152	10	2.0	1292	112	9.4	955	10:
2	152	12	2.2	1402	88	9.0	939	13
3	152	13	2.2	1376	80	9.2	948	17:
4	152	14	1.6	1272	51	6.5	836	13
...	...	...	...	...	...	...	...	..
9352	232	1	3.1	1314	-200	13.5	1101	47:
9353	232	2	2.4	1163	-200	11.4	1027	35:
9354	232	3	2.4	1142	-200	12.4	1063	29:
9355	232	4	2.1	1003	-200	9.5	961	23:
9356	232	5	2.2	1071	-200	11.9	1047	26:

9357 rows × 15 columns

In [156]:
```
df2 = df.drop('Date', axis=1)
```

In [157]:
```
df2 = df2.drop('Time', axis=1)
```

In [158]:
```
df2 = df2.drop('CO(GT)', axis=1)
```

In [159]:
```
df2 = df2.drop('C6H6(GT)', axis=1)
```

In [160]:

```
df2
```

Out[160]:

	PT08.S1(CO)	NMHC(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.
0	1360	150	1046	166	1056	113	
1	1292	112	955	103	1174	92	
2	1402	88	939	131	1140	114	
3	1376	80	948	172	1092	122	
4	1272	51	836	131	1205	116	
...	...	...	...	...	...	...	
9352	1314	-200	1101	472	539	190	
9353	1163	-200	1027	353	604	179	
9354	1142	-200	1063	293	603	175	
9355	1003	-200	961	235	702	156	
9356	1071	-200	1047	265	654	168	

9357 rows × 11 columns

In [161]:

```
df2 = df2.loc[(df2['T'] >= 24) & (df2['T'] < 24.2)]
df2
```

Out[161]:

	PT08.S1(CO)	NMHC(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.
1706	1379	-200	1312	267	683	179	
1987	1263	-200	1171	192	759	88	
2042	1058	-200	1134	-200	855	-200	
2080	891	-200	780	87	1110	68	
2190	938	-200	881	48	956	61	
2240	839	-200	725	36	1052	52	
2247	1123	-200	1088	157	746	92	
2343	1180	-200	1145	221	713	128	
2382	1013	-200	999	120	791	105	
2384	791	-200	658	30	1140	43	
2500	1004	-200	997	104	813	96	
2511	1277	-200	1300	331	608	136	
2574	870	-200	823	76	975	79	
2603	809	-200	628	27	1127	42	
2626	838	-200	632	24	1079	39	
2627	827	-200	656	55	1075	60	
2742	1131	-200	1087	164	705	131	
2766	1217	-200	1060	126	660	101	
2843	894	-200	709	46	997	53	
2975	1073	-200	1117	220	693	119	
3015	1211	-200	1290	311	606	156	
3030	1038	-200	1053	122	744	116	
3274	870	-200	581	15	1119	26	
3374	1281	-200	1469	287	491	136	
3421	948	-200	829	68	801	71	
3445	903	-200	648	19	975	21	
3581	1039	-200	991	93	685	74	
3609	886	-200	744	-200	869	-200	
3757	967	-200	826	68	821	61	
3990	904	-200	801	44	852	53	
4040	805	-200	637	24	1048	35	
4134	1053	-200	891	-200	690	-200	
4158	926	-200	769	-200	836	-200	
4161	832	-200	593	-200	1049	-200	
4232	944	-200	796	-200	784	-200	
4305	826	-200	504	-200	1250	-200	
4328	847	-200	634	-200	1012	-200	
4431	1202	-200	989	-200	775	-200	
4446	1251	-200	1055	-200	701	-200	

	PT08.S1(CO)	NMHC(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.
4531	1289	-200	1217	-200	599	-200	
4732	1111	-200	991	218	721	109	
5065	1747	-200	1709	515	399	128	
5092	1230	-200	990	151	652	71	
5127	1055	-200	795	103	807	51	
5518	1669	-200	1615	760	404	166	
5682	1426	-200	1445	667	477	151	
8904	1315	-200	1225	518	509	193	
9093	1273	-200	988	288	563	123	
9239	1271	-200	1183	403	493	179	
9259	970	-200	710	150	833	100	
9329	1000	-200	779	171	805	115	

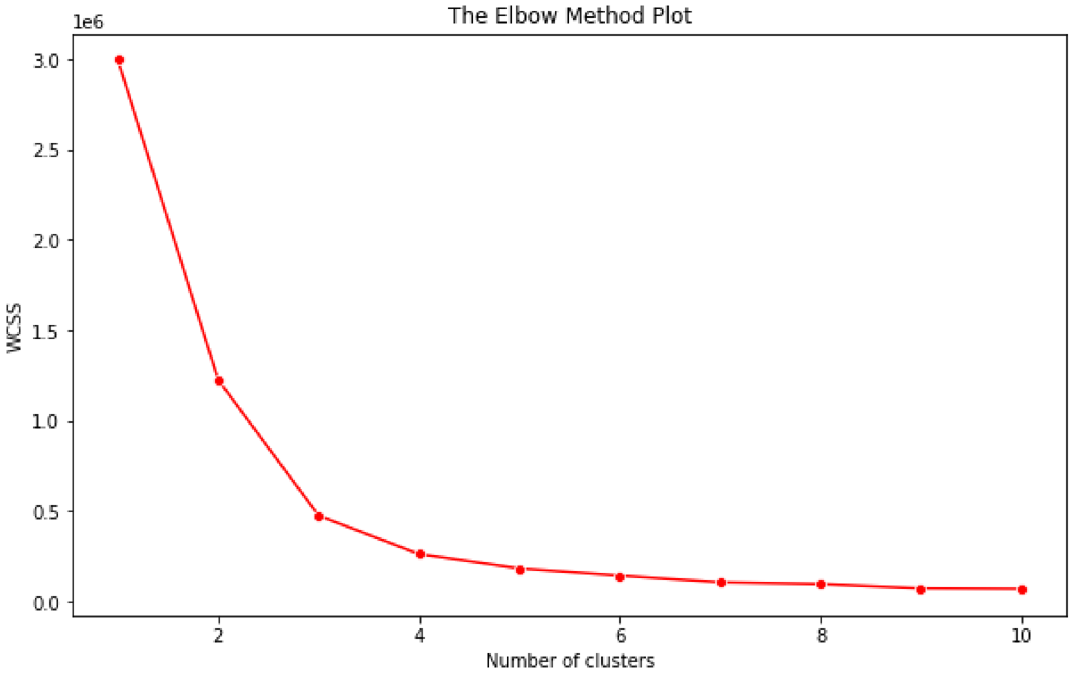
In [162]:

```
df2.drop(df2.index[df2['NOx(GT)'] == -200], inplace = True)
```

In [163]:

```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
X = df2.iloc[:, [3,4]].values
#X = X.reshape(-1,1)
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'random', random_state = 42)
    kmeans.fit(X)
    # inertia method returns wcss for that model
    wcss.append(kmeans.inertia_)
```

In [164]:
```
plt.figure(figsize=(10,6))
sns.lineplot(range(1, 11), wcss,marker='o',color='red')
plt.title('The Elbow Method Plot')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [165]:
```
kmeans = KMeans(n_clusters=3, init ='random', max_iter=300, n_init=10,r
kmeans.fit(X)
y_kmeans = kmeans.fit_predict(X)
```

In [167]:
```
df2['KMeans predicted value']=y_kmeans
```

In [168]:

```
plt.figure(figsize=(10,6))
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], c = 'brown', labe
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], c = 'green', labe
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], c = 'blue', label
#plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], c = 'purple', lal
#plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], c = 'yellow', lal

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:, 1]

plt.xlabel('CO(GT)')
plt.ylabel('PT08.S3 (NOx)')
plt.legend()
plt.title('Clustered Data')
```

Out[168]: Text(0.5, 1.0, 'Clustered Data')